Advanced HTML course outline

Introduction

Who I am

What we'll cover

Survey of class

- Mac vs. PC?
- Coding by hand? WYSIWYG? Other?
- Creating pages? Entire site? Serving a site?

Ridiculously quick review of basic HTML

HTML docs are text files

Tags surround this text to "mark it up" as particular items

- `<HTML></HTML>` surrounds everything
- `<HEAD></HEAD>` surrounds header info, and `<BODY></BODY>` the body
- Text that appears in a web page is tagged with `<P>` tag (no need to close with `</P>`)
- `<H1>` through `<H6>` mark text as headings
- `<B><I><STRONG><EMPHASIS>` tags format text appropriately
- `<PRE>` creates monospaced text that can be positioned with spaces
- `<HR>` inserts horizontal rules
- `<IMG SRC="blah.gif">` inserts graphics
- `Align graphics with <IMG SRC="blah.gif" ALIGN=TOP>`
    - `TOP, CENTER, BOTTOM` aligns to neighboring single line of text
    - `RIGHT, LEFT` allows text to wrap around graphic
    - `<BR CLEAR=ALL>` moves text past the graphic if you want to end the text wrap
- `<A HREF="http://www.site.com/">` specifies a hyperlink
- `<UL><LI>` tags create a bullet list, `<OL><LI>` creates a numbered list
- Create anchors via `<A NAME="theanchor">`
- Jump to it via `<A HREF="#theanchor">` if in the same document

Other HTML topics

Specifying graphic size

- `<IMG SRC="blah.gif" HEIGHT=50 WIDTH=50>`
- By giving dimensions of graphic, browsers can "leave a hole" for the graphic prior to loading it
- Browser can display the text faster, instead of waiting to determine how to layout the text
- You can also distort a graphic by using a larger/smaller number than the actual height or width
    - This is useful for Spacer GIFs—invisible GIFs used to move text over or down a specific amount
    - Can also be useful for graphics bars used as HRs

Relative vs. Absolute referencing—site organization

- Absolute bad, relative good.
- Absolute: `<IMG SRC="http://www.site.com/img/blah.gif">`
- Relative call is RELATIVE to where you are in the site's structure
- If you're at root level: `<IMG SRC="/img/blah.gif">`
- If you are in a folder at the same level as img...`<IMG SRC="../img/blah.gif">`
- ".." backs you out of your current folder and up the site's structure (just like DOS)
- Relative is better because it's more portable
  - If your web site changes names or locations, you don't have to change all your `IMG SRC` or `A HREF` calls
  - Less typing, therefore less room for error
  - It is ideal, as a result, to have a well-ordered web site
- Site organization
  - Ideally, plan this out prior to creating your site
  - Place often-used graphics into a single folder
  - Organize your documents logically
    - *By category*
    - *By month*
    - *By department*
    - *By project, tissue type… completely depends on the type of site*
  - Name documents and folders consistently

Image Maps (client side and server side)

Why do an imagemap? Why not do multiple graphics?

- Sometimes it's impossible (map of USA or of human body)
- A single imagemap loads faster than multiple buttons

Imagemaps link sections of a graphic to HREF links

Most people do client site image mapping now

- Virtually all browsers support it
- No need to run a special imagemap CGI on web server (less work for server)

Creating client-side imagemap…

- Draw your graphic, save as GIF or JPG
- Use `<IMG SRC="example.gif" USEMAP="#mymap">`
- Create map within `<MAP NAME="mymap"></MAP>` tags
- Each link corresponds to an AREA tag like so:
  ```
  <AREA SHAPE="circle" COORDS="x,y,r" HREF="link.html">
  <AREA SHAPE="rectangle" COORDS="x1,y1,x2,y2" HREF="link2.html">
  <AREA SHAPE="polygon" COORDS="x1,y1,x2,y2..." HREF="link3.html">
  ```
  `<AREA SHAPE="default" HREF=nohref>` (specifies no link if you click on an undefined area)
- Note that many tools are available for making this task MUCH easier
  - Standalone utilities are nice
  - I usually end up using WYSIWYG tools for this (FrontPage, PageMill, etc.)

Sound and multimedia

BRIEF Overview of different formats

- Sound
  - Traditional: .au, .wav, .aiff
  - Highly compressed: .ra (Real Audio), .mp3 (MPEG Layer 3)
  - Compact music format: .mid (MIDI format) (note: music only!)
- Animation
  - GIF animations (the easiest)
  - Macromedia (Shockwave (Director), Flash)
- Movies
  - QuickTime (.mov), Windows AVI format (.avi), RealVideo (.rm)

Embedding sound

- Various plug-ins support various ways of playing sound
  - Usually involve the `<EMBED SRC="blah">` tag
- Can always provide `<A HREF>` link to the media if you don't want or need to embed
- Streaming RealAudio instead of downloading RealAudio
  - Using `<A HREF>` link makes entire audio clip download
  - By using a "metafile", the RealAudio clip streams instead
    - *Create a text file with the http:// reference to RealAudio clip*
    - *Name it "whatever.ram" (for RealAudio Metafile)*
    - *Provide a link to the metafile rather than the clip itself*
    - *Server must be properly set up with correct MIME types*

Embedding animations or movies

- GIF animations: just `<IMG SRC>`! That's its advantage
- QuickTime or AVI movies: `<EMBED SRC="blah">`
- Various formats and plug-ins provide instructions for embedding that set controller options
- `<EMBED SRC="gal.mov" WIDTH=252 HEIGHT=267 CONTROLLER=TRUE AUTOPLAY=TRUE LOOP=TRUE>`

Tables and Frames

Tables

- `<TABLE BORDER="x" WIDTH="x" >`
- Each row of data has `<TR></TR>` tags; each column of data has `<TD></TD>` tags
- You can add `COLSPAN=n` if you want a cell to span multiple columns
- Add `BGCOLOR=#rrggbb` for table with a different background color
- The table's width and height can be specified via absolute (pixels) or relative (percentage) numbers
- You can nest tables in tables
- Complex sets of tables can be hard to follow
- WYSIWYG tools can help
  - HTML page editors
  - Dedicated table creation tools
  - Even Word/Excel/WordPerfect have table export tools
- There is a `HEIGHT=""` attribute that some WYSIWYG tools add
  - However, the contents of the table and cells usually determines (and overrides) the height attribute

Frames

- Frames are set up in much the same way as tables
- `<FRAMESET></FRAMESET>` surrounding the frame definition
- Then, each individual row or column is specified
  - Row height or column width can be specified as absolute (pixels) or relative (percent).
  - Can also specify `“*”:` fills in the remainder (useful when specifying absolute widths/heights)
- Sample horizontal frameset

```
<FRAMESET ROWS=“25%,75%”>
    <FRAME SRC=“doc1.html” NAME=“topframe”>
    <FRAME SRC=“doc2.html” NAME=“bottomframe”>
</FRAMESET>
```

- Similarly done for vertical sets of columns
- Or mix and match

```
<FRAMESET COLS=“20%,80%”>
   <FRAME SRC=“toc.html” NAME=“tableofcontents”>
   <FRAMESET ROWS=“75,*”>
      <FRAME SRC=“doc1.html” NAME=“headerframe”>
      <FRAME SRC=“doc2.html” NAME=“bodyframe”>
   </FRAMESET>
</FRAMESET>
```

- You then add to your `<A HREF> TARGET=“framename”` to load specific docs into specific frames
- Special `TARGET`s to know about (case sensitive!)
  - `TARGET=“_blank”` opens a new window
  - `TARGET=“_top”` reloads the current window (wipes out all frames)
  - `TARGET=“_self”` loads doc into the current frame
  - Can set `<BASE TARGET=“whatever”>` if you want most `<A HREF>` links to target a specific named frame
- To support frames-incapable browsers (or those who turn frames off), include `<NOFRAMES>` tags
  - Place at end of frameset definition
  - Insert HTML in `<BODY>` tags within the `<NOFRAMES>` tags
- A frameset-defining HTML document does not have `<BODY>`, unless it’s in `<NOFRAMES>`
- You can even load new framesets into specific named frames
  - This can get VERY confusing
  - Therefore, use useful names to avoid this confusion
- WYSIWYG editors can be useful in helping set up frames and framesets

BREAK

Return in 10 minutes!

Forms

Lets the user interact with your site

- User can request specific documents or pages
- Send information to your site
- Fill out online questionnaires
- Query databases
- And so on

Form tags are `<FORM ACTION="x" METHOD="POST">...</FORM>`

- All other form fields are entered within these tags

Before making a form, what will you do with form's results?

- Turns out that this is the hard part: creating the forms is easy
- Usually requires a CGI program running on your web server
  - This program will take the form info and process it
  - It then usually enters it into a database or does other calculations
  - Or it can email it to a given address
  - The CGI will determine the `ACTION=""` and `METHOD=""` parameters
- One workaround: `ACTION="mailto:youraddress@nih.gov"`
  - This mails the form's raw data to the given email address
  - However, it then requires a utility to "clean up" the form data or insert it into a database
  - These utilities (freeware, shareware) are available for Mac and PC

Form fields

- All fields have `NAME` attribute
  - This allows you to match up the fields to fields in a database
- Text input field: `<INPUT TYPE="text">`
  - You can specify a MAXLENGTH of text to enter
- Text areas: `<TEXTAREA></TEXTAREA>`
  - Can specify COLS and ROWS for the size, and whether text wraps or not
- Check boxes: `<INPUT TYPE="checkbox">`
  - Attribute of `CHECKED` puts a default check in the box
  - `VALUE` attribute sets what value gets sent if the box is checked
- Radio buttons: `<INPUT TYPE="radio">`
  - Radio buttons should be used when only one of multiple options should be selected
  - `NAME` all radio buttons belonging to the same group the same name, but different `VALUE`s
- Popup selections: `<SELECT></SELECT>`
  - Options go within `<OPTION></OPTION>` tags
  - Add `SELECTED` to pick a default selection
  - Add `VALUE=""` to determine what is sent when that option is selected
- List selections
  - Add `MULTIPLE` to the `SELECT` tag
  - Specify a `SIZE=` number of rows you want displayed
  - Allows user to select multiple options with CMD (Mac) or CTRL (PC) key
- Submit and Reset buttons
  - Pretty self explanatory

JavaScript

JavaScript is a programming language

- Unfortunately beyond the scope of this class
- And possibly this instructor! :-)
- I can demonstrate, however, its use and some basics

Basics on where to place scripts

- `<SCRIPT LANGUAGE="JavaScript"></SCRIPT>`
- Usually goes in `<HEAD>` and before `<BODY>`
- This should not display in other non-JavaScript browsers
- But in case it does, surround the script itself with HTML comment tags `<!--` and `-->`
- Most JavaScripts are triggered by a button option (`onClick="doSomething()"`)
- Some run when the page is loaded `<BODY onLoad="doSomething()">`
- Input and output are handled by forms, which are accessed in JavaScript by the form and field names... `if (document.formname.textfieldname.value == "yes")`

Script examples

- Mouse rollover
- Navigation pop-up

META tag information

What are META tags?

- HTML header information that provides information on the information in your page
- `<META>` found between `<HEAD>` and `</HEAD>`
- Can provide info to search engines on keywords, summary of your page, timed redirects to new pages, and "robot control."
- You can use multiple META tags in your page's header

The most useful META tags

- `<META HTTP-EQUIV="Refresh" CONTENT="10;URL=http://www.site.com">`
  will load the specified URL after 10 seconds
- `<META HTTP-EQUIV="Window-target" CONTENT="_top">`
  will attempt to clear existing framesets from a browser window before loading the page
- `<META NAME="keywords" CONTENT="advanced, HTML, Vargas, NIH, Baywatch">`
  provides indexing sites a list of keywords
- `<META NAME="description" CONTENT="Your description here!">`
  lets you write your own description to appear in search engines' online descriptions
- `<META NAME="robots" CONTENT="all | none | index | noindex | follow | nofollow">`
  specifies how robots should index the page
  - "none" allows no indexing OR following of links on page
  - "noindex" disallows indexing, but allows following links
  - "nofollow" disallows following links, but allows indexing

Cascading style sheets

Newest 4.0 browsers support CSS

- Unfortunately, they support them to differing degrees
- A “write once run anywhere” style sheet is not yet available
- However, you can use style sheets invisibly to browsers that don’t support CSS
- If your document requires a high degree of formatting, consider using PDF instead of HTML

Brief introduction to CSS use

- Most will use CSS to change font
- You can change font style without CSS by using the `FACE=“”` option of the `<FONT></FONT>` tag
  - `<FONT FACE=“Helvetica, Arial, sans-serif”>`
  - Problems with this approach
    - *To change font, you must search/replace throughout all of your HTML files*
    - *Less flexibility*
    - *Creates cluttered HTML code*
- Enter the Cascading Style Sheet
  - You can use styles in a number of ways
    - *Applied to specific parts of your document*
    - *Creating new “style names” to apply to your document*
    - *Redefining existing HTML tags to display differently*
    - *I will emphasize this last technique*
  - You can also insert style definitions in different ways
    - *Placed into the* `<HEAD>` *of a document*
    - *Or, defined in its own document and called by documents*
    - *I’ll demonstrate both of these techniques*
- Defining styles in a given document
  - In `<HEAD>`, insert `<STYLE TYPE=“text/css”>`
  - Use `<!--` and `-->` tags to surround style definitions (same as JavaScript trick)
  - Redefine tags by listing them and entering new definition in braces...

```
H4 { font-family:Verdana, Arial, Helvetica, sans-serif; font-size:medium;
font-weight:bold; color:#330066; background-color:#66CC00 }
P { font-family:Palatino, serif; margin-left: 10%; margin-right: 10% }
BODY {background: URL(tile.gif)}
```

  - Close off definitions with `</STYLE>`
  - Use these HTML tags in your text and view with CSS-capable browser
  - Advantage to using HTML tags is that they appear correctly in non-CSS capable browsers too
- Defining styles in a separate document and linking
  - Create a text document called “whatever.css”
  - Insert just the style definitions listed above... no `<STYLE>` tags
  - In the main document, insert into `<HEAD>` section...

```
<LINK REL=stylesheet HREF=“whatever.css” type=“text/css”>
```

  - Make sure the .css document and main document are in the same folder (or use relative `HREF`)
- Overriding styles?
  - Use the `<SPAN STYLE=“”></SPAN>` tags for local overrides
  - Insert new `</STYLE>` definitions for document overrides
- Style sheet formatting options?
  - Not all browsers support all options
  - Most allow you to set font, color, and size